Лабораторная работа № 4.1

Тема: Изучение аффинных преобразований на плоскости.

Задание.

В мировой системе координат (МСК) XOY заданы два отрезка прямых:

1. **AB** с координатами $A(2 \quad 12)$ и $B(5 \quad 9)$;
2. **CD** с координатами $C(2 \quad 3)$ и $D(5 \quad 6)$;

Пусть **P** – точка пересечения отрезков **AB** и **CD** или их продолжений, или одного их них с продолжением другого.

Система из двух отрезков **AB** и **CD** поворачивается вокруг точки **P** на угол 180° против часовой стрелки.

Вычислить:

1. координаты точки **P** в системе координат XOY ;
2. новые координаты точек **A**, **B**, **C**, **D** в системе координат XOY .

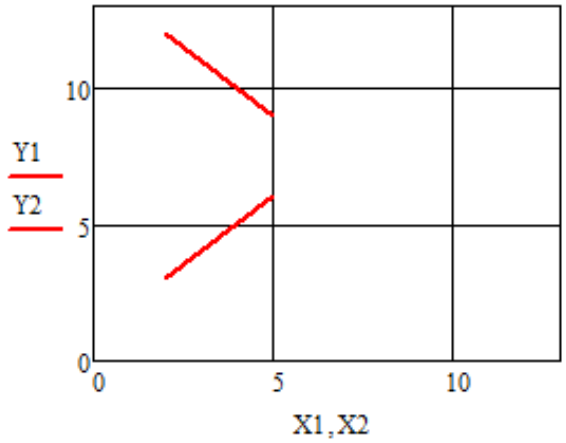
Отобразить в системе координат XOY ***разными цветами:***

1. исходное положение отрезков прямых **AB** и **CD** (красный);
2. положение отрезков прямых **AB** и **CD** после поворота (синий);
3. точку **P**, вокруг которой выполняется вращение (черный).

Запишем условие задания

$X1 := \begin{pmatrix} 2 \\ 5 \end{pmatrix}$ $Y1 := \begin{pmatrix} 12 \\ 9 \end{pmatrix}$ - координаты отрезка AB

$X2 := \begin{pmatrix} 2 \\ 5 \end{pmatrix}$ $Y2 := \begin{pmatrix} 3 \\ 6 \end{pmatrix}$ - координаты отрезка CD



Найдем точку пересечения прямых

Given

$x + y = 14$ - уравнение прямой AB

$x - y = -1$ - уравнение прямой CD

$\text{Find}(x,y) \rightarrow \begin{pmatrix} \frac{13}{2} \\ \frac{15}{2} \end{pmatrix}$ $Px := \frac{13}{2}$ $Py := \frac{15}{2}$

Найдем новые координаты точек A,B,C,D после поворота на угол 180°

$P := \begin{pmatrix} X1_0 & X1_1 & X2_0 & X2_1 \\ Y1_0 & Y1_1 & Y2_0 & Y2_1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$ - матрица координат

$Ts(dx,dy) := \begin{pmatrix} 1 & 0 & -dx \\ 0 & 1 & -dy \\ 0 & 0 & 1 \end{pmatrix}$ - смещение начала системы координат

$$Rs(\phi) := \begin{pmatrix} \cos(\phi) & \sin(\phi) & 0 \\ -\sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$ - поворот системы координат на угол ϕ

$\phi := 180$ $\quad \phi\_rad := \frac{\phi \cdot \pi}{180}$ - угол в радианах

Сместим систему координат в точку P

$$A := Ts(Px, Py) \cdot P = \begin{pmatrix} -4.5 & -1.5 & -4.5 & -1.5 \\ 4.5 & 1.5 & -4.5 & -1.5 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

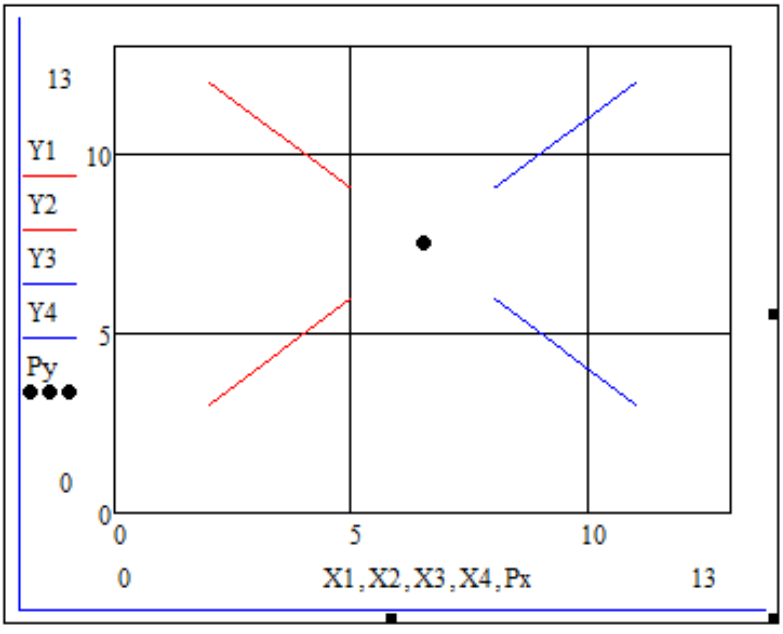
Повернем систему координат на 180 градусов

$$B := Rs(\phi\_rad) \cdot A = \begin{pmatrix} 4.5 & 1.5 & 4.5 & 1.5 \\ -4.5 & -1.5 & 4.5 & 1.5 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

Снова сместим систему координат в исходное положение

$$C := Ts(-Px, -Py) \cdot B = \begin{pmatrix} 11 & 8 & 11 & 8 \\ 3 & 6 & 12 & 9 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

$X3 := \begin{pmatrix} 11 \\ 8 \end{pmatrix}$ $\quad Y3 := \begin{pmatrix} 3 \\ 6 \end{pmatrix}$ $\qquad\qquad X4 := \begin{pmatrix} 11 \\ 8 \end{pmatrix}$ $\quad Y4 := \begin{pmatrix} 12 \\ 9 \end{pmatrix}$



Лабораторная работа № 4.2

Тема: Изучение аффинных преобразований на плоскости на примере объекта «Планеты»

I. Реализовать функции аффинных преобразований и добавить их в библиотеку Libgraph

CMatrix CreateTranslate2D (double dx, double dy)

// Формирует матрицу для преобразования координат **объекта** при его смещении
// на dx по оси X и на dy по оси Y в фиксированной системе координат
// --- ИЛИ ---
// Формирует матрицу для преобразования координат объекта при смещении начала
// системы координат на -dx оси X и на -dy по оси Y при фиксированном положении объекта

CMatrix CreateRotate2D (double fi)

// Формирует матрицу для преобразования координат **объект**а при его повороте
// на угол fi (при fi>0 против часовой стрелки) в фиксированной системе координат
// --- ИЛИ ---
// Формирует матрицу для преобразования координат объекта при повороте начала
// системы координат на угол -fi при фиксированном положении объекта
// fi - угол в градусах

II. Реализовать объект «Планеты» реализовать как класс CSunSystem.

```
class CSunSystem
{
        CRect Sun;          // Прямоугольник Солнца
        CRect Earth;                // Прямоугольник Земли
        CRect Moon;             // Прямоугольник Луны
        CRect EarthOrbit;       // Прямоугольник, описанный вокруг орбиты Земли
        CRect MoonOrbit;        // Прямоугольник, описанный вокруг орбиты Луны
        CMatrix ECoords;        // Текущие координаты Земли в СК Солнца ( x,y,1)
        CMatrix MCoords;        // Текущие координаты Луны в СК Земли ( x,y,1)
        CMatrix MCoords1;       // Текущие координаты Луны в СК Солнца ( x,y,1)
        CMatrix PM;             // Матрица поворота для луны
        CMatrix PE;             // Матрица поворота для Земли
```

```
        CRect RW;           // Прямоугольник в окне
        CRectD RS;          // Прямоугольник области в МСК
        double wEarth;              // Угловая скорость Земли относительно
Солнца, град./сек.
        double wMoon;       // Угловая скорость Земли относительно Солнца,
град./сек.
        double fiE;         // Угловое положение Земли в системе кординат
Солнца, град
        double dfiE;          // Угол поворота Земли за время dt.
        double fiM;         // Угловое положение Луны в системе кординат
Земли, град
        double dfiM;                // Угол поворота Земли за время dt

        double dt;            // Интервал дискретизации, сек.
 public:
        CSunSystem();
        void SetDT(double dtx){dt=dtx;};  // Установка интервала дискретизации
        void SetNewCoords();              // Вычисляет новые координаты
планет
        void GetRS(CRectD& RSX);     // Возвращает область графика в
мировой СК
        CRect GetRW(){return RW;};   // Возвращает область графика в окне
        void Draw(CDC& dc);           // Рисование без самостоятельного
пересчета координат
};
```

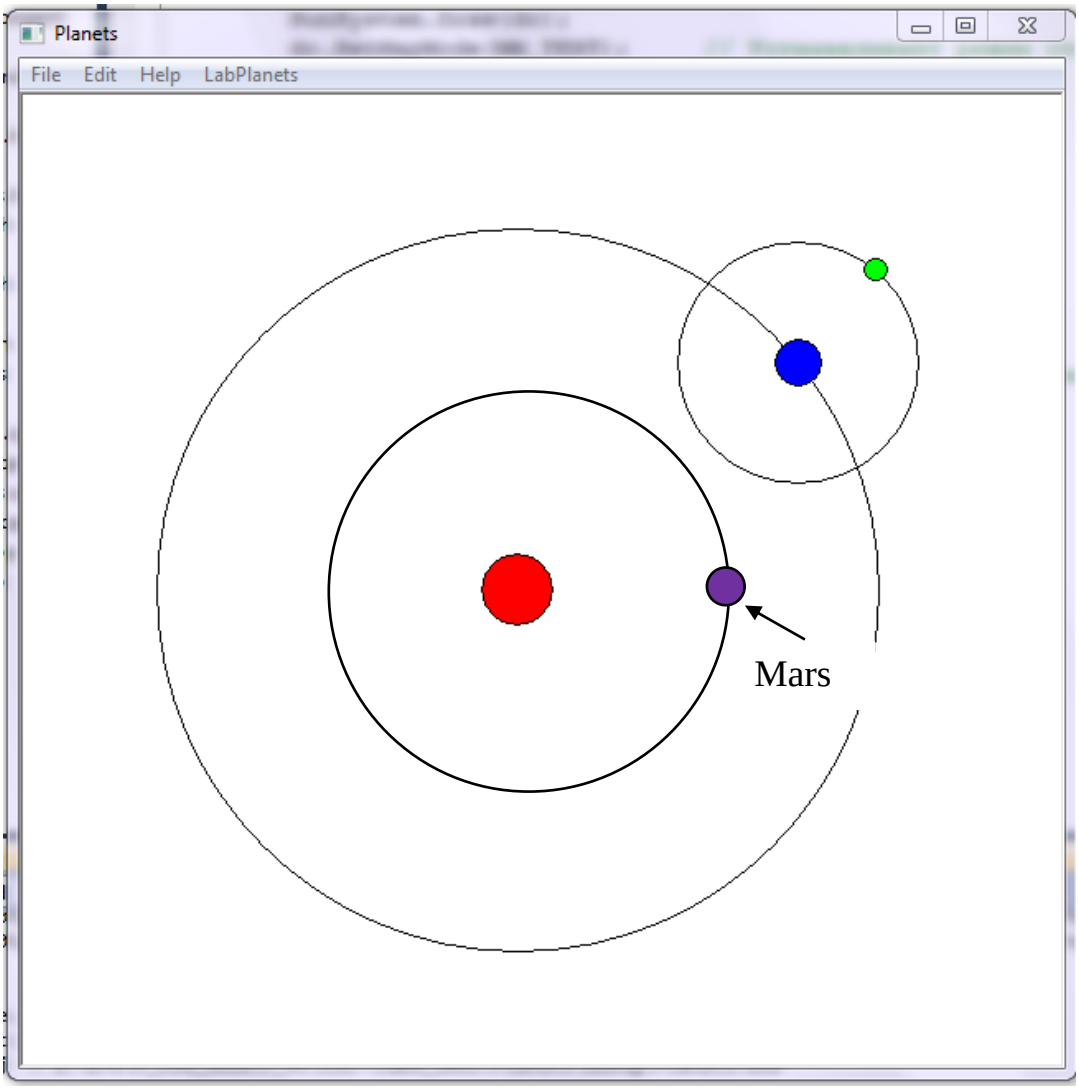
Добавить в класс CSunSystem планету Mars, орбита которой находится между Солнцем и Землей. Новая планета вращается в направлении, противоположном вращению Земли.
Рисование выполнять в режиме MM_**ANISOTROPIC**. Толщина линии – **1пиксел**.

Сценарий работы

- После запуска приложения на экране появляется пустое окно.
- После выбора пункта меню «LabPlanets►Planets» на экране появляется статичное изображение планет, размер и положение которых определяются в соответствии с параметрами, заданными в конструкторе по умолчанию.
- После двойного щелчка ЛКМ (левая клавиша мыши) в области окна планеты приходят в движение.

- После двойного щелчка ПКМ (правая клавиша мыши) в области окна движение планет прекращается.

Размеры окна устанавливаются **по габаритам системы планет и не должны изменяться.**



Аффинные преобразования

«Аффи́нное преобразование , иногда афинное преобразование — отображение плоскости или пространства в себя, при котором параллельные прямые переходят в параллельные прямые, пересекающиеся — в пересекающиеся, скрещивающиеся — в скрещивающиеся».

Под преобразованием объекта будем понимать изменение координат точек, принадлежащих этому объекту при изменении его положения в некоторой системе координат. Пусть в пространстве задана система

координат XYZ и точка M(x, y,z) принадлежащая некоторому объекту. Объект перемещается из одной точки пространства в другую путем ряда последовательных смещений и поворотов относительно своего исходного положения. При этом система координат остается неподвижной. Необходимо определить новые координаты точки M(x ′ , y ′ ,z ′) (объекта) в СК XYZ В общем случае старые координаты точки (x, y,z) и новые (x ′ , y ′ ,z) связаны соотношениями (9.1) – (9.9). Подчеркнём лишь, что в данном случае выражения (9.1) – (9.9) описывают взаимосвязь между старыми и новыми координатами точки M при изменении ее положения в одной и той же системе координат. Рассмотрим частные случаи аффинных преобразований координат объектов.

Смещение объекта вдоль координатных осей (рис.9.7)

$$\begin{cases} x' = x + \Delta x \\ y' = y + \Delta y, \\ z' = z + \Delta z \end{cases} \qquad (9.34)$$

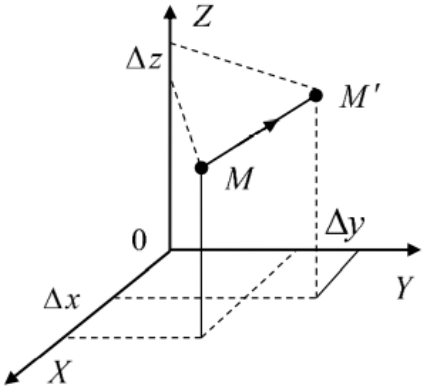


Рис. 9.7

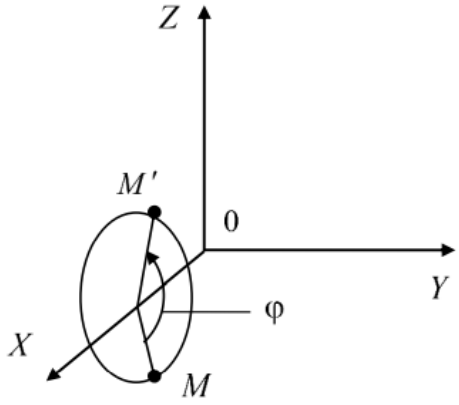


Рис. 9.8

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \qquad (9.35)$$

Обратное преобразование

Обратное преобразование соответствует перемещению объекта в противоположном направлении. Соответствующие выражения для преобразования координат могут быть получены из (9.35) – (9.38) путем соответствующей замены в них $(x', y', z') \to (x, y, z)$ и $(\Delta x, \Delta y, \Delta z) \to (-\Delta x, -\Delta y, -\Delta z)$.

Приведем выражение для матрицы преобразования.

$$A' = \left(T^o\right)^{-1} = \left[T^o(\Delta x, \Delta y, \Delta z)\right]^{-1} = \begin{pmatrix} 1 & 0 & 0 & -\Delta x \\ 0 & 1 & 0 & -\Delta y \\ 0 & 0 & 1 & -\Delta z \\ 0 & 0 & 0 & 1 \end{pmatrix} = T^o(-\Delta x, -\Delta y, -\Delta z) \quad (9.38)$$

Растяжение – сжатие объекта

$$\begin{cases} x' = k_x x \\ y' = k_y y \\ z' = k_z z \end{cases}, \quad (9.39)$$

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} k_x & 0 & 0 & 0 \\ 0 & k_y & 0 & 0 \\ 0 & 0 & k_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}, \quad (9.40)$$

Обратное преобразование

Обратное преобразование соответствует повороту объекта вокруг оси X в противоположном направлении. Соответствующие выражения для преобразования координат могут быть получены из (9.45) – (9.48) путем соответствующей замены в них (x ′ , y ′ ,z ′) → (x, y,z) и φ → – φ

Приведем выражение для матрицы преобразования.

$$A' = \left(R_X^o\right)^{-1} = \left(R_X^o(\varphi)\right)^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi & 0 \\ 0 & -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = R_X^o(-\varphi) \qquad (9.48)$$

Поворот объекта вокруг оси Y на угол φ (рис. 9.9).

$$\begin{cases} x' = x\cos\varphi + z\sin\varphi \\ y' = y \\ z' = -x\sin\varphi + z\cos\varphi \end{cases} \qquad (9.49)$$

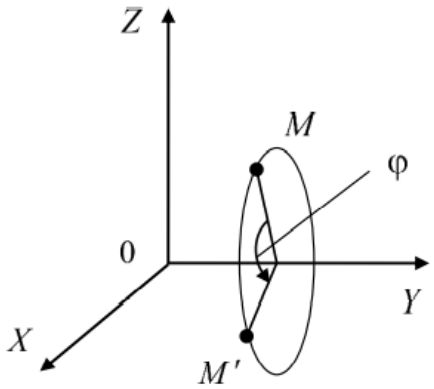


Рис. 9.9

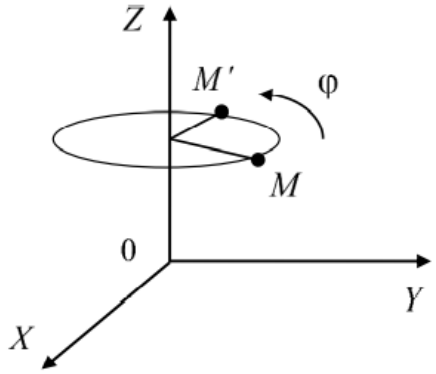
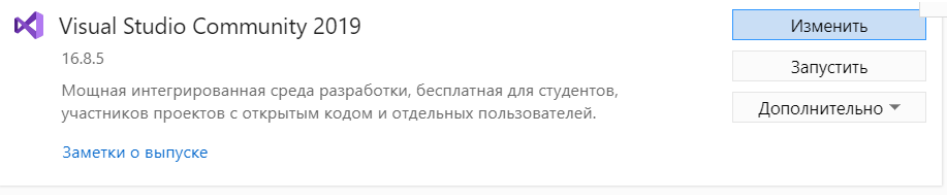


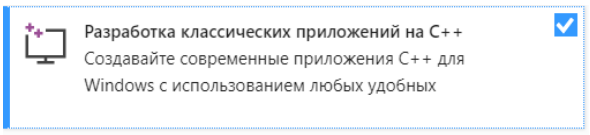
Рис. 9.10

Для выполнения этой лабораторной работой, вам требуется установить библиотеку MFC.

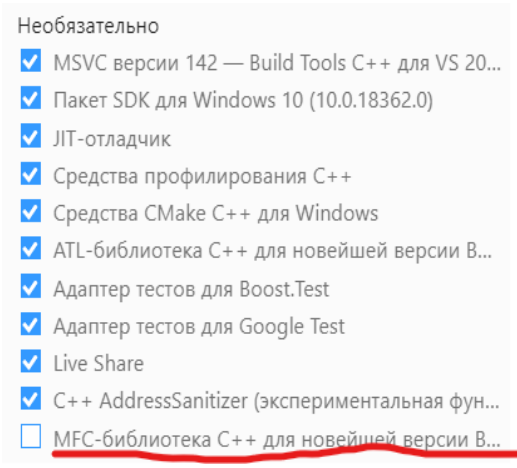
Заходим в Visual Studio Installer. Для добавления новых компонентов нажмите кнопку “Изменить”.



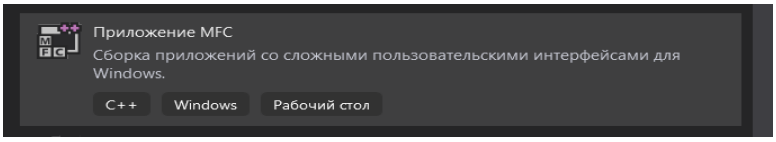
Выбираем пункт Разработка классических приложений на C++.



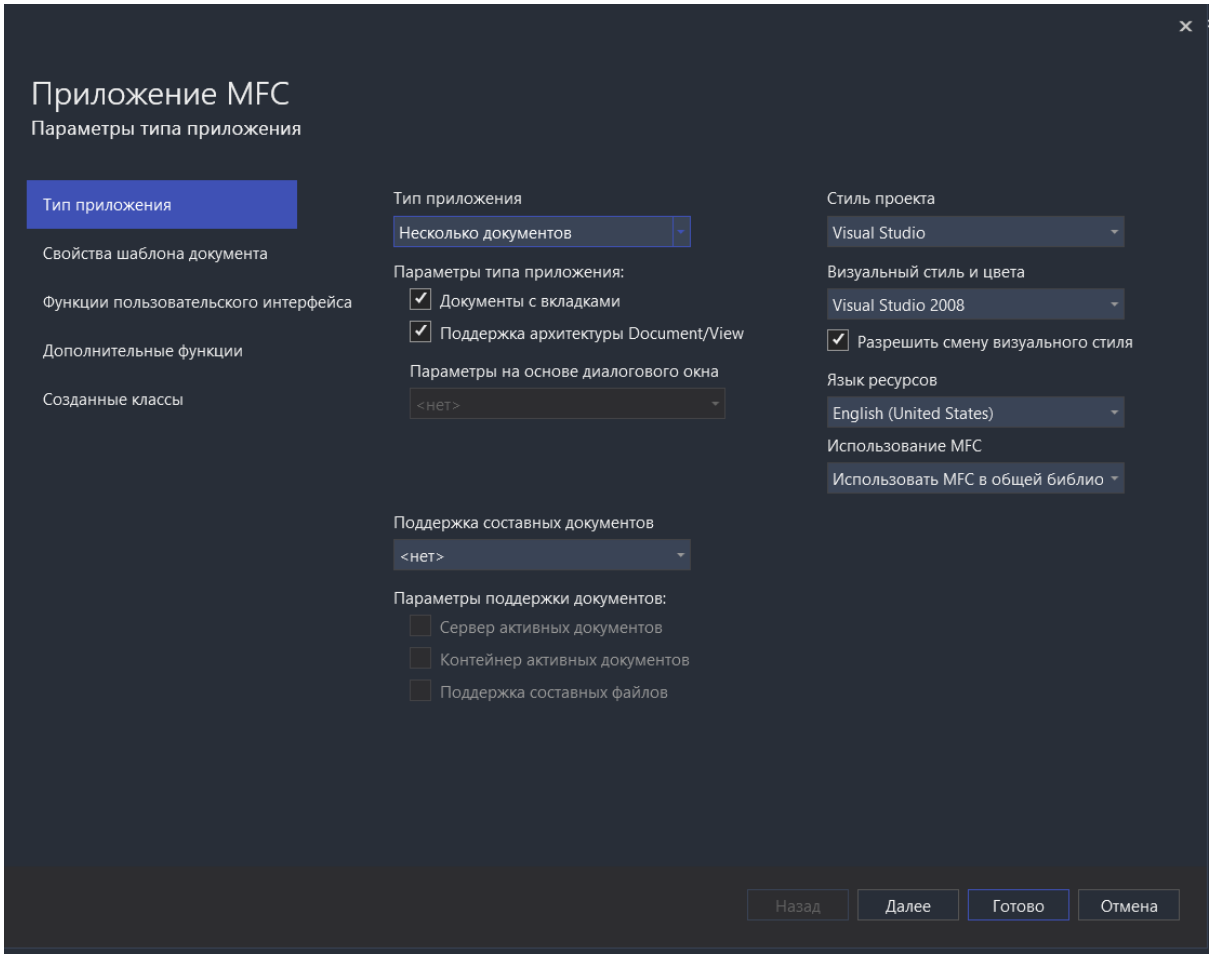
Далее в меню пакетов выбираем MFC-библиотеку C++.

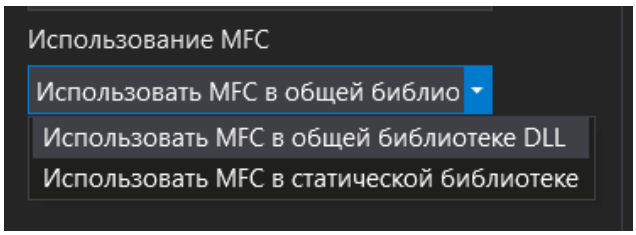


Создаём приложение MFC



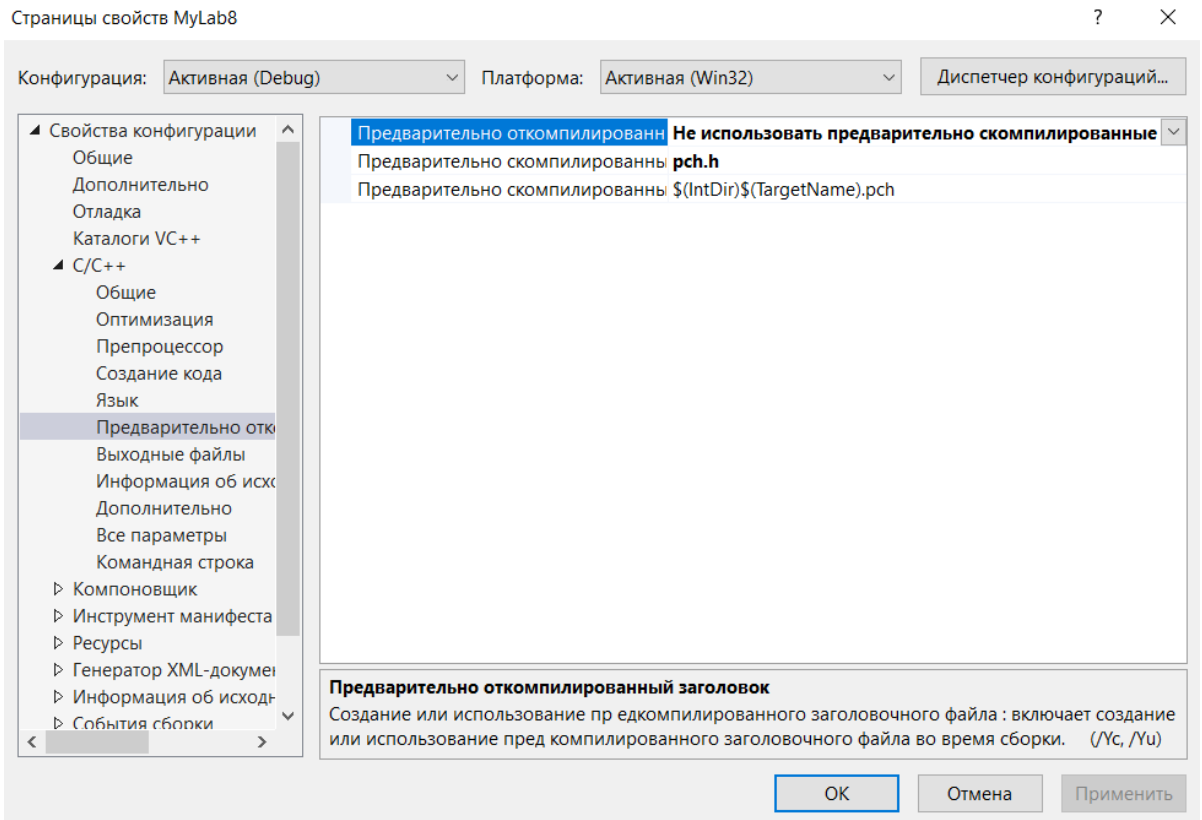
В настройках убеждаемся, что библиотека MFC будет использоваться в общей DLL





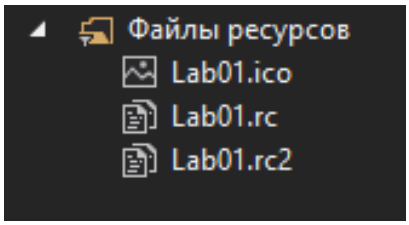
После удаления мусора, заходим в свойства проекта. Тыкаем на C/C++ и выбираем Предварительно откомпилированный или как там не вижу по скрину. И ставим что мы НЕ БУДЕМ использовать этот pch.h. Мы будем юзать stdafx для мерджина.

Теперь заходим в свойства проекта, C/C++, Предварительно откомпилированные и выбрать “Не использовать предварительно скомпилированные заголовки”.



Чтобы начать работу над проектом, требуется удалить все созданные, при создании приложения, файлы.

Вместе с этой лабораторной работой будет архив с нужными ресурсами. Их всего 3. Закидываем эти ресурсы в папку res, и добавляем в наш проект. Получится вот так.



Их не надо менять, код, написанный далее. Он будет забинжен под эти ресурсы.

Далее создадим файл stdafx.h и stdafx.cpp, в котором будет служебная информация а также подключения нужных нам заголовочных файлов.

stdafx.h:

```
// stdafx.h: включите файл для добавления стандартных системных файлов
//или конкретных файлов проектов, часто используемых,
// но редко изменяемых

#pragma once

#ifndef VC_EXTRALEAN
#define VC_EXTRALEAN            // Исключите редко используемые компоненты из заголовков
Windows
#endif

#include "targetver.h"

#define _ATL_CSTRING_EXPLICIT_CONSTRUCTORS      // некоторые конструкторы CString будут
явными

// отключает функцию скрытия некоторых общих и часто пропускаемых предупреждений MFC
#define _AFX_ALL_WARNINGS

#include <afxwin.h>         // основные и стандартные компоненты MFC
#include <afxext.h>         // расширения MFC


#include <fstream>
#include "CMatrix.h"
#include "LibPlanets.h"



#ifndef _AFX_NO_OLE_SUPPORT
#include <afxdtctl.h>           // поддержка MFC для типовых элементов управления
Internet Explorer 4
#endif
#ifndef _AFX_NO_AFXCMN_SUPPORT
#include <afxcmn.h>             // поддержка MFC для типовых элементов управления
Windows
#endif // _AFX_NO_AFXCMN_SUPPORT
```

```
#include <afxcontrolbars.h>     // поддержка MFC для лент и панелей управления




#ifdef _UNICODE
#if defined _M_IX86
#pragma comment(linker,"/manifestdependency:\"type='win32'
name='Microsoft.Windows.Common-Controls' version='6.0.0.0' processorArchitecture='x86'
publicKeyToken='6595b64144ccf1df' language='*'\"")
#elif defined _M_X64
#pragma comment(linker,"/manifestdependency:\"type='win32'
name='Microsoft.Windows.Common-Controls' version='6.0.0.0' processorArchitecture='amd64'
publicKeyToken='6595b64144ccf1df' language='*'\"")
#else
#pragma comment(linker,"/manifestdependency:\"type='win32'
name='Microsoft.Windows.Common-Controls' version='6.0.0.0' processorArchitecture='*'
publicKeyToken='6595b64144ccf1df' language='*'\"")
#endif
#endif
```

Большинство заголовочных файлов будет подчеркнуто красным. Реализацию этих файлов напишем позже.

```
//{{NO_DEPENDENCIES}}
// Включаемый файл, созданный в Microsoft Visual C++.
// Используется в Lab03.rc
//
#define IDD_ABOUTBOX                    100
#define IDP_OLE_INIT_FAILED             100
#define IDR_MAINFRAME                   128
#define IDR_Lab01TYPE                   130
#define ID_LAB_PLANETS                32771

// Следующие стандартные значения для новых объектов
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        312
#define _APS_NEXT_COMMAND_VALUE         32776
#define _APS_NEXT_CONTROL_VALUE         1000
#define _APS_NEXT_SYMED_VALUE           310
#endif
#endif
```

targetver.h

```
#pragma once

// Включение SDKDDKVer.h обеспечивает определение самой последней доступной платформы
Windows.

// Если требуется выполнить сборку приложения для предыдущей версии Windows, включите
WinSDKVer.h и
```

```
// задайте для макроопределения _WIN32_WINNT значение поддерживаемой платформы перед
вхождением SDKDDKVer.h.

#include <SDKDDKVer.h>
```

MainFrm.h

```
// MainFrm.h: интерфейс класса CMainFrame
//

#pragma once
#include "ChildView.h"

class CMainFrame : public CFrameWnd
{

public:
	CMainFrame() noexcept;
protected:
	DECLARE_DYNAMIC(CMainFrame)

// Атрибуты
public:

// Операции
public:

// Переопределение
public:
	virtual BOOL PreCreateWindow(CREATESTRUCT& cs);
	virtual BOOL OnCmdMsg(UINT nID, int nCode, void* pExtra, AFX_CMDHANDLERINFO*
pHandlerInfo);

// Реализация
public:
	virtual ~CMainFrame();
#ifdef _DEBUG
	virtual void AssertValid() const;
	virtual void Dump(CDumpContext& dc) const;
#endif

protected:  // встроенные члены панели элементов управления
	CStatusBar        m_wndStatusBar;
	CChildView    m_wndView;

// Созданные функции схемы сообщений
protected:
	afx_msg int OnCreate(LPCREATESTRUCT lpCreateStruct);
	afx_msg void OnSetFocus(CWnd *pOldWnd);
	DECLARE_MESSAGE_MAP()

};
```

MainFrm.cpp

```
// MainFrm.cpp: реализация класса CMainFrame
//

#include "header.h"
#include "Lab04.h"

#include "MainFrm.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

// CMainFrame

IMPLEMENT_DYNAMIC(CMainFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
	ON_WM_CREATE()
	ON_WM_SETFOCUS()
END_MESSAGE_MAP()

static UINT indicators[] =
{
	ID_SEPARATOR,           // индикатор строки состояния
	ID_INDICATOR_CAPS,
	ID_INDICATOR_NUM,
	ID_INDICATOR_SCRL,
};

// Создание или уничтожение CMainFrame

CMainFrame::CMainFrame() noexcept
{
	// TODO: добавьте код инициализации члена
}

CMainFrame::~CMainFrame()
{
}

int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
	if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
		return -1;

	// создать представление для размещения рабочей области рамки
	if (!m_wndView.Create(nullptr, nullptr, AFX_WS_DEFAULT_VIEW, CRect(0, 0, 0, 0),
this, AFX_IDW_PANE_FIRST, nullptr))
	{
		TRACE0("Не удалось создать окно представлений\n");
		return -1;
	}
```

```
	if (!m_wndStatusBar.Create(this))
	{
		TRACE0("Не удалось создать строку состояния\n");
		return -1;	// не удалось создать
	}
	m_wndStatusBar.SetIndicators(indicators, sizeof(indicators)/sizeof(UINT));

	return 0;
}

BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
	if( !CFrameWnd::PreCreateWindow(cs) )
		return FALSE;
	// TODO: изменить класс Window или стили посредством изменения
	//  CREATESTRUCT cs

	cs.style = WS_OVERLAPPED | WS_SYSMENU | WS_BORDER; // задаём вид окна

	cs.dwExStyle &= ~WS_EX_CLIENTEDGE;
	cs.lpszClass = AfxRegisterWndClass(0);
	return TRUE;
}

// Диагностика CMainFrame

#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
	CFrameWnd::AssertValid();
}

void CMainFrame::Dump(CDumpContext& dc) const
{
	CFrameWnd::Dump(dc);
}
#endif //_DEBUG


// Обработчики сообщений CMainFrame

void CMainFrame::OnSetFocus(CWnd* /*pOldWnd*/)
{
	// передача фокуса окну представления
	m_wndView.SetFocus();
}

BOOL CMainFrame::OnCmdMsg(UINT nID, int nCode, void* pExtra, AFX_CMDHANDLERINFO*
pHandlerInfo)
{
	// разрешить ошибки в представлении при выполнении команды
	if (m_wndView.OnCmdMsg(nID, nCode, pExtra, pHandlerInfo))
		return TRUE;
```

```
	// в противном случае выполняется обработка по умолчанию
	return CFrameWnd::OnCmdMsg(nID, nCode, pExtra, pHandlerInfo);
}
```

CMatrix.h

```
#pragma once
#include <fstream>
using namespace std;
#ifndef CMATRIXH
# define CMATRIXH 1
class CMatrix
{
	double **array;
	int n_rows;			// Число строк
	int n_cols;			// Число столбцов
public:
	CMatrix();			// Конструктор по умолчанию (1 на 1)

	CMatrix(int, int);		// Конструктор
	CMatrix(int);			//  Конструктор -вектора (один
столбец)
	CMatrix(const CMatrix&);		// Конструктор копирования
	CMatrix(ifstream &file)
	{
		int r = 0;
		int c = 0;
		file >> r;
		file >> c;

		// Nrow - число строк
		// Ncol - число столбцов

		n_rows = r;
		n_cols = c;
		array = new double*[n_rows];
		for (int i = 0; i < n_rows; i++) array[i] = new
double[n_cols];

		for (int i = 0; i < n_rows; i++)
			for (int j = 0; j < n_cols; j++) file >> array[i][j];
	}
	~CMatrix();
	double &operator()(int, int);		// Выбор элемента матрицы по
индексу
	double &operator()(int);		// Выбор элемента вектора по
индексу
	CMatrix operator-();			// Оператор "-"
	CMatrix operator=(const CMatrix&); //	Оператор "Присвоить":
M1=M2
	CMatrix operator*(CMatrix&);		// Оператор "Произведение":
M1*M2
	CMatrix operator+(CMatrix&);	// Оператор "+": M1+M2
	CMatrix operator-(CMatrix&);	// Оператор "-": M1-M2
	CMatrix operator+(double);		// Оператор "+": M+a
	CMatrix operator-(double);		// Оператор "-": M-a
```

```
	friend std::ostream& operator<<(std::ostream& os, CMatrix& matrix)
	{
		os << matrix.rows() << ' ' << matrix.cols() << '\n';

		for (int i = 0; i < matrix.rows(); i++)
		{
			for (int k = 0; k < matrix.cols(); k++)
			{
				os << matrix(i, k) << " ";
			}
			os << '\n';
		}
		return os;
	}
	int rows()const { return n_rows; };   // Возвращает число строк
	int cols()const { return n_cols; };    // Возвращает число строк
	CMatrix Transp();
//Возвращаетматрицу,транспонированную к текущей
	CMatrix GetRow(int);                    //   Возвращает   строку   по
номеру
	CMatrix GetRow(int, int, int);
	CMatrix GetCol(int);                    //   Возвращает   столбец   по
номеру
	CMatrix GetCol(int, int, int);
	CMatrix RedimMatrix(int, int);          // Изменяет размер матрицы с
уничтожением данных
	CMatrix RedimData(int, int);           // Изменяет размер матрицы с
сохранением данных,
                                                        //которые
можно сохранить
	CMatrix RedimMatrix(int);             // Изменяет размер матрицы с
уничтожением данных
	CMatrix RedimData(int);                // Изменяет размер матрицы с
сохранением данных,
                                                        //которые
можно сохранить
	double MaxElement();                      //   Максимальный   элемент
матрицы
	double MinElement();                      //   Минимальный   элемент
матрицы
};


#endif
```

CMatrix.cpp

```
#include "stdafx.h"
#include "CMatrix.h"

CMatrix::CMatrix()
{
	n_rows = 1;
	n_cols = 1;
	array = new double*[n_rows];
	for (int i = 0; i < n_rows; i++) array[i] = new double[n_cols];
```

```
	for (int i = 0; i < n_rows; i++)
		for (int j = 0; j < n_cols; j++) array[i][j] = 0;
}

//---------------------------------------------------------------------------
CMatrix::CMatrix(int Nrow, int Ncol)
// Nrow - число строк
// Ncol - число столбцов
{
	n_rows = Nrow;
	n_cols = Ncol;
	array = new double*[n_rows];
	for (int i = 0; i < n_rows; i++) array[i] = new double[n_cols];
	for (int i = 0; i < n_rows; i++)
		for (int j = 0; j < n_cols; j++) array[i][j] = 0;
}

//---------------------------------------------------------------------------
CMatrix::CMatrix(int Nrow)  //Вектор
// Nrow - число строк
{
	n_rows = Nrow;
	n_cols = 1;
	array = new double*[n_rows];
	for (int i = 0; i < n_rows; i++) array[i] = new double[n_cols];
	for (int i = 0; i < n_rows; i++)
		for (int j = 0; j < n_cols; j++) array[i][j] = 0;
}
//---------------------------------------------------------------------------
CMatrix::~CMatrix()
{
	for (int i = 0; i < n_rows; i++) delete array[i];
	delete array;
}

//---------------------------------------------------------------------------
double &CMatrix::operator()(int i, int j)
// i - номер строки
// j - номер столбца
{
	if ((i > n_rows - 1) || (j > n_cols - 1))     //  проверка выхода за диапазон
	{
		TCHAR* error = _T("CMatrix::operator(int,int): выход индекса за границу диапазона ");
		MessageBox(NULL, error, _T("Ошибка"), MB_ICONSTOP);
		exit(1);
	}
	return array[i][j];
}

//---------------------------------------------------------------------------
double &CMatrix::operator()(int i)
// i - номер строки для вектора
{
	if (n_cols > 1)     //  Число столбцов больше одного
	{
```

```
                char* error = "CMatrix::operator(int): объект не вектор - число столбцов больше 1 ";
                MessageBox(NULL, (LPCWSTR)error, L"Ошибка", MB_ICONSTOP);
                exit(1);
        }
        if (i > n_rows - 1)     //  проверка выхода за диапазон
        {
                TCHAR* error = TEXT("CMatrix::operator(int): выход индекса за границу диапазона ");
                MessageBox(NULL, error, TEXT("Ошибка"), MB_ICONSTOP);
                exit(1);
        }
        return array[i][0];
}
//---------------------------------------------------------------------------
CMatrix CMatrix::operator-()
// Оператор -M
{
        CMatrix Temp(n_rows, n_cols);
        for (int i = 0; i < n_rows; i++)
                for (int j = 0; j < n_cols; j++) Temp(i, j) = -array[i][j];
        return Temp;
}

//---------------------------------------------------------------------------
CMatrix CMatrix::operator+(CMatrix& M)
// Оператор M1+M2
{
        int bb = (n_rows == M.rows()) && (n_cols == M.cols());
        if (!bb)
        {
                char* error = "CMatrix::operator(+): несоответствие размерностей матриц ";
                MessageBox(NULL, (LPCWSTR)error, L"Ошибка", MB_ICONSTOP);
                exit(1);
        }
        CMatrix Temp(*this);
        for (int i = 0; i < n_rows; i++)
                for (int j = 0; j < n_cols; j++) Temp(i, j) += M(i, j);
        return Temp;
}

//---------------------------------------------------------------------------
CMatrix CMatrix::operator-(CMatrix& M)
// Оператор M1-M2
{
        int bb = (n_rows == M.rows()) && (n_cols == M.cols());
        if (!bb)
        {
                char* error = "CMatrix::operator(-): несоответствие размерностей матриц ";
                MessageBox(NULL, (LPCWSTR)error, L"Ошибка", MB_ICONSTOP);
                exit(1);
        }
        CMatrix Temp(*this);
        for (int i = 0; i < n_rows; i++)
                for (int j = 0; j < n_cols; j++) Temp(i, j) -= M(i, j);
        return Temp;
}
```

```
//---------------------------------------------------------------------------
CMatrix CMatrix::operator*(CMatrix& M)
// Умножение на матрицу M
{
	double sum;
	int nn = M.rows();
	int mm = M.cols();
	CMatrix Temp(n_rows, mm);
	if (n_cols == nn)
	{
		for (int i = 0; i < n_rows; i++)
			for (int j = 0; j < mm; j++)
			{
				sum = 0;
				for (int k = 0; k < n_cols; k++) sum += (*this)(i, k)*M(k, j);
				Temp(i, j) = sum;
			}
	}
	else
	{
		TCHAR* error = TEXT("CMatrix::operator*: несоответствие размерностей матриц ");
		MessageBox(NULL, error, TEXT("Ошибка"), MB_ICONSTOP);
		exit(1);
	}
	return Temp;
}

//---------------------------------------------------------------------------
CMatrix CMatrix::operator=(const CMatrix& M)
// Оператор присваивания M1=M
{
	if (this == &M) return *this;
	int nn = M.rows();
	int mm = M.cols();
	if ((n_rows == nn) && (n_cols == mm))
	{
		for (int i = 0; i < n_rows; i++)
			for (int j = 0; j < n_cols; j++) array[i][j] = M.array[i][j];
	}
	else   // для ошибки размерностей
	{
		TCHAR* error = TEXT("CMatrix::operator=: несоответствие размерностей матриц");
		MessageBox(NULL, error, TEXT("Ошибка"), MB_ICONSTOP);
		exit(1);
	}
	return *this;
}

//---------------------------------------------------------------------------
CMatrix::CMatrix(const CMatrix &M) // Конструктор копирования
{
	n_rows = M.n_rows;
	n_cols = M.n_cols;
	array = new double*[n_rows];
	for (int i = 0; i < n_rows; i++) array[i] = new double[n_cols];
```

```
	for (int i = 0; i < n_rows; i++)
		for (int j = 0; j < n_cols; j++) array[i][j] = M.array[i][j];
}

//---------------------------------------------------------------------------
CMatrix CMatrix::operator+(double x)
// Оператор M+x, где M - матрица, x - число
{
	CMatrix Temp(*this);
	for (int i = 0; i < n_rows; i++)
		for (int j = 0; j < n_cols; j++) Temp(i, j) += x;
	return Temp;
}

//---------------------------------------------------------------------------
CMatrix CMatrix::operator-(double x)
// Оператор M+x, где M - матрица, x - число
{
	CMatrix Temp(*this);
	for (int i = 0; i < n_rows; i++)
		for (int j = 0; j < n_cols; j++) Temp(i, j) -= x;
	return Temp;
}



//---------------------------------------------------------------------------
CMatrix CMatrix::Transp()
// Возвращает матрицу,транспонированную к (*this)
{
	CMatrix Temp(n_cols, n_rows);
	for (int i = 0; i < n_cols; i++)
		for (int j = 0; j < n_rows; j++) Temp(i, j) = array[j][i];
	return Temp;
}

//---------------------------------------------------------------------------
CMatrix CMatrix::GetRow(int k)
// Возвращает строку матрицы по номеру k
{
	if (k > n_rows - 1)
	{
		char* error = "CMatrix::GetRow(int k): параметр k превышает число строк ";
		MessageBox(NULL, (LPCWSTR)error, L"Ошибка", MB_ICONSTOP);
		exit(1);
	}
	CMatrix M(1, n_cols);
	for (int i = 0; i < n_cols; i++)M(0, i) = (*this)(k, i);
	return M;
}
//---------------------------------------------------------------------------
CMatrix CMatrix::GetRow(int k, int n, int m)
// Возвращает подстроку из строки матрицы с номером k
// n - номер первого элемента в строке
// m - номер последнего элемента в строке
```

```
{
	int b1 = (k >= 0) && (k < n_rows);
	int b2 = (n >= 0) && (n <= m);
	int b3 = (m >= 0) && (m < n_cols);
	int b4 = b1 && b2&&b3;
	if (!b4)
	{
		char* error = "CMatrix::GetRow(int k,int n, int m):ошибка в параметрах ";
		MessageBox(NULL, (LPCWSTR)error, L"Ошибка", MB_ICONSTOP);
		exit(1);
	}
	int nCols = m - n + 1;
	CMatrix M(1, nCols);
	for (int i = n; i <= m; i++)M(0, i - n) = (*this)(k, i);
	return M;
}

//---------------------------------------------------------------------------
CMatrix CMatrix::GetCol(int k)
// Возвращает столбец матрицы по номеру k
{
	if (k > n_cols - 1)
	{
		char* error = "CMatrix::GetCol(int k): параметр k превышает число столбцов ";
		MessageBox(NULL, (LPCWSTR)error, L"Ошибка", MB_ICONSTOP);
		exit(1);
	}
	CMatrix M(n_rows, 1);
	for (int i = 0; i < n_rows; i++)M(i, 0) = (*this)(i, k);
	return M;
}
//---------------------------------------------------------------------------
CMatrix CMatrix::GetCol(int k, int n, int m)
// Возвращает подстолбец из столбца матрицы с номером k
// n - номер первого элемента в столбце
// m - номер последнего элемента в столбце
{
	int b1 = (k >= 0) && (k < n_cols);
	int b2 = (n >= 0) && (n <= m);
	int b3 = (m >= 0) && (m < n_rows);
	int b4 = b1 && b2&&b3;
	if (!b4)
	{
		char* error = "CMatrix::GetCol(int k,int n, int m):ошибка в параметрах ";
		MessageBox(NULL, (LPCWSTR)error, L"Ошибка", MB_ICONSTOP);
		exit(1);
	}
	int nRows = m - n + 1;
	CMatrix M(nRows, 1);
	for (int i = n; i <= m; i++)M(i - n, 0) = (*this)(i, k);
	return M;
}
//---------------------------------------------------------------------------
CMatrix CMatrix::RedimMatrix(int NewRow, int NewCol)
// Изменяет размер матрицы с уничтожением данных
```

```
// NewRow - новое число строк
// NewCol - новое число столбцов
{
	for (int i = 0; i < n_rows; i++) delete array[i];
	delete array;
	n_rows = NewRow;
	n_cols = NewCol;
	array = new double*[n_rows];
	for (int i = 0; i < n_rows; i++) array[i] = new double[n_cols];
	for (int i = 0; i < n_rows; i++)
		for (int j = 0; j < n_cols; j++) array[i][j] = 0;
	return (*this);
}

//---------------------------------------------------------------------------
CMatrix CMatrix::RedimData(int NewRow, int NewCol)
// Изменяет размер матрицы с сохранением данных, которые можно сохранить
// NewRow - новое число строк
// NewCol - новое число столбцов
{
	CMatrix Temp = (*this);
	this->RedimMatrix(NewRow, NewCol);
	int min_rows = Temp.rows() < (*this).rows() ? Temp.rows() : (*this).rows();
	int min_cols = Temp.cols() < (*this).cols() ? Temp.cols() : (*this).cols();
	for (int i = 0; i < min_rows; i++)
		for (int j = 0; j < min_cols; j++) (*this)(i, j) = Temp(i, j);
	return (*this);
}


//---------------------------------------------------------------------------
CMatrix CMatrix::RedimMatrix(int NewRow)
// Изменяет размер матрицы с уничтожением данных
// NewRow - новое число строк
// NewCol=1
{
	for (int i = 0; i < n_rows; i++) delete array[i];
	delete array;
	n_rows = NewRow;
	n_cols = 1;
	array = new double*[n_rows];
	for (int i = 0; i < n_rows; i++) array[i] = new double[n_cols];
	for (int i = 0; i < n_rows; i++)
		for (int j = 0; j < n_cols; j++) array[i][j] = 0;
	return (*this);
}

//---------------------------------------------------------------------------
CMatrix CMatrix::RedimData(int NewRow)
// Изменяет размер матрицы с сохранением данных, которые можно сохранить
// NewRow - новое число строк
// NewCol=1
{
	CMatrix Temp = (*this);
	this->RedimMatrix(NewRow);
```

```
		int min_rows = Temp.rows() < (*this).rows() ? Temp.rows() : (*this).rows();
		for (int i = 0; i < min_rows; i++)(*this)(i) = Temp(i);
		return (*this);
	}
	//--------------------------------------------------------------------------
	double CMatrix::MaxElement()
	// Максимальное значение элементов матрицы
	{
		double max = (*this)(0, 0);
		for (int i = 0; i < (this->rows()); i++)
			for (int j = 0; j < (this->cols()); j++) if ((*this)(i, j) > max) max = (*this)(i, j);
		return max;
	}

	//--------------------------------------------------------------------------
	double CMatrix::MinElement()
	// Минимальное значение элементов матрицы
	{
		double min = (*this)(0, 0);
		for (int i = 0; i < (this->rows()); i++)
			for (int j = 0; j < (this->cols()); j++) if ((*this)(i, j) < min) min = (*this)(i, j);
		return min;
	}
```

LibPlanets.h:

```
#ifndef LIBPLANETS
#define LIBPLANETS 1
const double pi = 3.14159;


struct CSizeD
{
	double cx;
	double cy;
};
//-----------------------------------------------------------------------------
struct CRectD
{
	double left;
	double top;
	double right;
	double bottom;
	CRectD() { left = top = right = bottom = 0; };
	CRectD(double l, double t, double r, double b);
	void SetRectD(double l, double t, double r, double b);
	CSizeD SizeD();		// Возвращает размеры(ширина, высота) прямоугольника
};
//-----------------------------------------------------------------------------

CMatrix CreateTranslate2D(double dx, double dy);
CMatrix CreateRotate2D(double fi);
CMatrix SpaceToWindow(CRectD& rs, CRect& rw);
void SetMyMode(CDC& dc, CRectD& RS, CRect& RW);
```

```
/////////////////  class CSunSystem ////////////////////////////////////////////

class CSunSystem
{
	CRect Sun;			// Прямоугольник Солнца
	CRect Earth;		// Прямоугольник Земли
	CRect Moon;			// Прямоугольник Луны
	CRect Mars;
	CRect EarthOrbit;	// Прямоугольник, описанный вокруг орбиты Земли
	CRect MoonOrbit;	// Прямоугольник, описанный вокруг орбиты Луны
	CRect MarsOrbit;
	CMatrix ECoords;	// Текущие координаты Земли в СК Солнца ( x,y,1)
	CMatrix MCoords;	// Текущие координаты Луны в СК Земли ( x,y,1)
	CMatrix VCoords;
	CRect RW;			// Прямоугольник в окне
	CRectD RS;			// Прямоугольник области в МСК
	double wEarth;		// Угловая скорость Земли относительно Солнца, град./сек.
	double wMoon;		// Угловая скорость Земли относительно Солнца, град./сек.
	double wMars;
	double fiE;			// Угловое положение Земли в системе координат Солнца, град
	double fiM;			// Угловое положение Луны в системе координат Земли, град
	double fiV;

	double dt;			// Интервал дискретизации, сек.
public:
	CSunSystem();
	void SetDT(double dtx) { dt = dtx; };	// Установка интервала дискретизации
	void SetNewCoords();				// Вычисляет новые координаты планет
	void GetRS(CRectD& RSX);			//  Возвращает область графика в
мировой СК
	CRect GetRW() { return RW; };		// Возвращает область графика в окне
	void Draw(CDC& dc);					//  Рисование без
самостоятельного пересчета координат
};


#endif
```

LibPlanets.cpp:

```
#include "stdafx.h"

CRectD::CRectD(double l, double t, double r, double b)
{
	left = l;
	top = t;
	right = r;
	bottom = b;
}
//--------------------------------------------------------------------------
void CRectD::SetRectD(double l, double t, double r, double b)
{
	left = l;
	top = t;
	right = r;
	bottom = b;
```

```
}

//---------------------------------------------------------------------------
CSizeD CRectD::SizeD()
{
      CSizeD cz;
      cz.cx = fabs(right - left);   // Ширина прямоугольной области
      cz.cy = fabs(top - bottom);   // Высота прямоугольной области
      return cz;
}

//---------------------------------------------------------------------------

CMatrix CreateTranslate2D(double dx, double dy)
// Формирует матрицу для преобразования координат объекта при его смещении
// на dx по оси X и на dy по оси Y в фиксированной системе координат
// --- ИЛИ ---
// Формирует матрицу для преобразования координат объекта при смещении начала
// системы координат на -dx оси X и на -dy по оси Y при фиксированном положении объекта
{
      CMatrix TM(3, 3);
      TM(0, 0) = 1; TM(0, 2) = dx;
      TM(1, 1) = 1;  TM(1, 2) = dy;
      TM(2, 2) = 1;
      return TM;
}

//---------------------------------------------------------------------------
CMatrix CreateRotate2D(double fi)
// Формирует матрицу для преобразования координат объекта при его повороте
// на угол fi (при fi>0 против часовой стрелки)в фиксированной системе координат
// --- ИЛИ ---
// Формирует матрицу для преобразования координат объекта при повороте начала
// системы координат на угол -fi при фиксированном положении объекта
// fi - угол в градусах
{
      double fg = fmod(fi, 360.0);
      double ff = (fg / 180.0)*pi; // Перевод в радианы
      CMatrix RM(3, 3);
      RM(0, 0) = cos(ff); RM(0, 1) = -sin(ff);
      RM(1, 0) = sin(ff);  RM(1, 1) = cos(ff);
      RM(2, 2) = 1;
      return RM;
}


//---------------------------------------------------------------------------

CMatrix SpaceToWindow(CRectD& RS, CRect& RW)
// Возвращает матрицу пересчета координат из мировых в оконные
// RS - область в мировых координатах - double
// RW - область в оконных координатах - int
{
      CMatrix M(3, 3);
      CSize sz = RW.Size();    // Размер области в ОКНЕ
```

```
      int dwx = sz.cx;       // Ширина
      int dwy = sz.cy;       // Высота
      CSizeD szd = RS.SizeD(); // Размер области в МИРОВЫХ координатах

      double dsx = szd.cx;    // Ширина в мировых координатах
      double dsy = szd.cy;    // Высота в мировых координатах

      double kx = (double)dwx / dsx;   // Масштаб по x
      double ky = (double)dwy / dsy;   // Масштаб по y

      M(0, 0) = kx;  M(0, 1) = 0;    M(0, 2) = (double)RW.left - kx * RS.left;

      M(1, 0) = 0;   M(1, 1) = -ky;  M(1, 2) = (double)RW.bottom + ky * RS.bottom;
      M(2, 0) = 0;   M(2, 1) = 0;    M(2, 2) = 1;
      return M;
}

//-----------------------------------------------------------------------------

void SetMyMode(CDC& dc, CRectD& RS, CRect& RW)
// Устанавливает режим отображения MM_ANISOTROPIC и его параметры
// dc - ссылка на класс CDC MFC
// RS -  область в мировых координатах - int
// RW -      Область в оконных координатах - int
{
      double dsx = RS.right - RS.left;
      double dsy = RS.top - RS.bottom;
      double xsL = RS.left;
      double ysL = RS.bottom;

      int dwx = RW.right - RW.left;
      int dwy = RW.bottom - RW.top;
      int xwL = RW.left;
      int ywH = RW.bottom;

      dc.SetMapMode(MM_ANISOTROPIC);
      dc.SetWindowExt((int)dsx, (int)dsy);
      dc.SetViewportExt(dwx, -dwy);
      dc.SetWindowOrg((int)xsL, (int)ysL);
      dc.SetViewportOrg(xwL, ywH);
}

////////////////////// CLASS CSunSystem ////////////////////////////////

CSunSystem::CSunSystem()            //  Конструктор по умолчанию
{
      double rS = 30, rE = 20, rM = 10, rV = 15;       // Радиус Солнца, Земли, Луны
      double RoE = 10 * rS, RoM = 5 * rE, RoV = 6*rS; // Радиус орбиты Земли и Луны
      double d = RoE + RoM + rM + RoV;              // Половина диаметра системы
      RS.SetRectD(-d, d, d, -d);                          //  Область системы в мировых
координатах
      RW.SetRect(0, 0, 600, 600);                         // Область в окне
      Sun.SetRect(-rS, rS, rS, -rS);                      //  Прямоугольник солнца - для
рисования круга
      Earth.SetRect(-rE, rE, rE, -rE);               // Прямоугольник Земли - для рисования
```

```
круга
	Moon.SetRect(-rM, rM, rM, -rM);                    //  Прямоугольник  Луны  -  для
рисования круга
	Mars.SetRect(-rV, rV, rV, -rV);
	EarthOrbit.SetRect(-RoE, RoE, RoE, -RoE);//  Прямоугольник  орбиты  Земли  -  для
рисования круга
	MoonOrbit.SetRect(-RoM, RoM, RoM, -RoM); // Прямоугольник орбиты Луны - для рисования
круга
	MarsOrbit.SetRect(-RoV, RoV, RoV, -RoV);
	fiE = 0;               // Угловое положение Земли в системе кординат Солнца, град
	fiM = 0;               // Угловое положение Луны в системе кординат Земли, град
	fiV = 1;
	wEarth = 5;            // Угловая скорость Земли в системе кординат Солнца, град/сек.
	wMoon = 50;            // Угловая скорость Луны в системе кординат Земли, град/сек.
	wMars = -8;
	dt = 0.1;
	MCoords.RedimMatrix(3);
	ECoords.RedimMatrix(3);
	VCoords.RedimMatrix(3);
}

void CSunSystem::SetNewCoords()
//Вычисляет новые координаты планет в СК Солнца через интервал времени dt
{
	double RoM = (MoonOrbit.right - MoonOrbit.left) / 2;// Радиус орбиты Луны
	double ff = (fiM / 180.0)*pi;                       //   Радианы   -   угловое
положение Луны в СК Земли
	double x = RoM * cos(ff);                           //   x   -   начальная
координата Луны в СК Земли
	double y = RoM * sin(ff);                           //   y   -   начальная
координата Луны в СК Земли
	MCoords(0) = x;   MCoords(1) = y;   MCoords(2) = 1;
	fiM += wMoon * dt;
	CMatrix P = CreateRotate2D(fiM);                    // Матрица поворота против
часовой стрелки Луны
	MCoords = P * MCoords;

	double RoE = (EarthOrbit.right - EarthOrbit.left) / 2;// Радиус орбиты Земли
	ff = (fiE / 180.0)*pi;                              // Радианы - угловое
положение Земли в СК Солнца
	x = RoE * cos(ff);                                  //      x      -
начальная координата Земли в СК Солнца
	y = RoE * sin(ff);                                  //      y      -
начальная координата Земли в СК Солнца
	ECoords(0) = x;   ECoords(1) = y;   ECoords(2) = 1;
	P = CreateTranslate2D(x,y);
	MCoords = P * MCoords;

	fiE += wEarth * dt;
	P = CreateRotate2D(fiE);                            //  Матрица  поворота
против часовой стрелки Луны и Земли
	MCoords = P * MCoords;
	ECoords = P * ECoords;

	double RoV = (MarsOrbit.right - MarsOrbit.left) / 2;
```

```
	ff = (fiV / 180.0)*pi;
	x = RoV * cos(ff);
	y = RoV * sin(ff);
	VCoords(0) = x;
	VCoords(1) = y;
	VCoords(2) = 1;

	fiV += wMars * dt;
	P = CreateRotate2D(fiV);
	VCoords = P * VCoords;
}


void CSunSystem::Draw(CDC& dc)
{
	CBrush SBrush, EBrush, MBrush, VBrush, *pOldBrush;
	CRect R;

	SBrush.CreateSolidBrush(RGB(255, 0, 0));
	EBrush.CreateSolidBrush(RGB(0, 0, 255));
	MBrush.CreateSolidBrush(RGB(0, 255, 0));
	VBrush.CreateSolidBrush(RGB(128,64,64));

	// Рисуем орбиты
	dc.SelectStockObject(NULL_BRUSH);		// Белая кисть - не надо соэдавать
	dc.Ellipse(EarthOrbit);				// Орбита Земли
	dc.Ellipse(MarsOrbit);

	int d = MoonOrbit.right;				// Радиус орбиты Луны
	R.SetRect(ECoords(0) - d, ECoords(1) + d, ECoords(0) + d, ECoords(1) - d);
	dc.Ellipse(R);						// Орбита Луны

// Рисуем Солнце
	pOldBrush = dc.SelectObject(&SBrush);	// Цвет Солнца
	dc.Ellipse(Sun);  // Солнце

// Рисуем Землю
	d = Earth.right;					// Радиус Земли
	R.SetRect(ECoords(0) - d, ECoords(1) + d, ECoords(0) + d, ECoords(1) - d);
	dc.SelectObject(&EBrush);				// Цвет Земли
	dc.Ellipse(R);    // Земля

// Рисуем Луну
	d = Moon.right;					// Радиус Луны
	R.SetRect(MCoords(0) - d, MCoords(1) + d, MCoords(0) + d, MCoords(1) - d);
	dc.SelectObject(&MBrush);				// Цвет Луны
	dc.Ellipse(R);    // Луна

	d = Mars.right;
	R.SetRect(VCoords(0) - d, VCoords(1) + d, VCoords(0) + d, VCoords(1) - d);
	dc.SelectObject(&VBrush);
	dc.Ellipse(R);

	dc.SelectObject(pOldBrush);			//Восстанавливаем контекст по
pOldBrush
```

```
}


void CSunSystem::GetRS(CRectD& RSX)
// RS - структура, куда записываются параметры области графика
{
	RSX.left = RS.left;
	RSX.top = RS.top;
	RSX.right = RS.right;
	RSX.bottom = RS.bottom;
}
```

ChildView.h:

```
// ChildView.h: интерфейс класса CChildView
//


#pragma once


// Окно CChildView

class CChildView : public CWnd
{
// Создание
public:
	CChildView();

// Атрибуты
public:
	int IsData;                    // Флаг готовности данных
	int dT_Timer;                  // Интервал времени для таймера
	CRect RW;                      // Область в окне
	CRectD RS;                     // Область в мировых координатах
	CSunSystem SunSystem;   // Объект (конструктор по умолчанию)
// Операции
public:

// Переопределение
	protected:
	virtual BOOL PreCreateWindow(CREATESTRUCT& cs);

// Реализация
public:
	virtual ~CChildView();

	// Созданные функции схемы сообщений
protected:
	afx_msg void OnPaint();
	DECLARE_MESSAGE_MAP()
public:
	// действия при выборе пункта меню
	afx_msg void OnLabplanetsPlanets();
	afx_msg void OnTimer(UINT_PTR nIDEvent);
	afx_msg void OnLButtonDblClk(UINT nFlags, CPoint point);
	afx_msg void OnRButtonDblClk(UINT nFlags, CPoint point);
};
```

ChildView.cpp:

```
// ChildView.cpp: реализация класса CChildView
//

#include "stdafx.h"
#include "Lab04.h"
#include "ChildView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif


// CChildView

CChildView::CChildView()
{
	IsData = 0;
}

CChildView::~CChildView()
{
}

// Реализация карты сообщений
BEGIN_MESSAGE_MAP(CChildView, CWnd)
	ON_WM_PAINT()
	// сообщения меню выбора
	ON_COMMAND(ID_LAB_PLANETS, &CChildView::OnLabplanetsPlanets)
	ON_WM_TIMER()
	ON_WM_LBUTTONDBLCLK()
	ON_WM_RBUTTONDBLCLK()
END_MESSAGE_MAP()



// Обработчики сообщений CChildView

BOOL CChildView::PreCreateWindow(CREATESTRUCT& cs)
{
	if (!CWnd::PreCreateWindow(cs))
		return FALSE;

	cs.dwExStyle |= WS_EX_CLIENTEDGE;
	cs.style &= ~WS_BORDER;
	cs.lpszClass = AfxRegisterWndClass(CS_HREDRAW|CS_VREDRAW|CS_DBLCLKS,
		::LoadCursor(nullptr, IDC_ARROW), reinterpret_cast<HBRUSH>(COLOR_WINDOW+1),
nullptr);

	return TRUE;
}

void CChildView::OnPaint()
{
	CPaintDC dc(this); // контекст устройства для рисования
```

```
	if (IsData == 1)
	{
		SunSystem.GetRS(RS);
		RW = SunSystem.GetRW();
		SetMyMode(dc, RS, RW);		// Устанавливает режим отображения
MM_ANISOTROPIC
		SunSystem.Draw(dc);
		dc.SetMapMode(MM_TEXT);		// Устанавливает режим отображения MM_TEXT

	}
}

	void CChildView::OnLabplanetsPlanets()	// PLANETS
	{
		SunSystem.SetDT(0);			// Для начального состояния
		SunSystem.SetNewCoords();
		SunSystem.SetDT(0.3);		//Обновление координат через 0.3 сек.
		dT_Timer = 100;				// Миллисекунд
		IsData = 1;
		Invalidate();

	}

	void CChildView::OnTimer(UINT_PTR nIDEvent)
	{
		SunSystem.SetNewCoords();
		Invalidate();
		CWnd::OnTimer(nIDEvent);

	}

	void CChildView::OnLButtonDblClk(UINT nFlags, CPoint point)
	{
		SetTimer(1, dT_Timer, NULL);
		CWnd::OnLButtonDblClk(nFlags, point);
	}

	void CChildView::OnRButtonDblClk(UINT nFlags, CPoint point)
	{
		KillTimer(1);
		CWnd::OnRButtonDblClk(nFlags, point);
	}
```

И реализуем файл запуска. Название файла должно соответствовать названию проекта.

Lab04.h:

```
// Lab04.h: основной файл заголовка для приложения Lab04
//
#pragma once

#ifndef __AFXWIN_H__
	#error "включить stdafx.h до включения этого файла в PCH"
```

```
#endif

#include "resource.h"       // основные символы


// CLab01App:
// Сведения о реализации этого класса: Lab01.cpp
//

class CLab01App : public CWinApp
{
public:
        CLab01App() noexcept;

// Переопределение
public:
        virtual BOOL InitInstance();
        virtual int ExitInstance();

// Реализация

public:
        afx_msg void OnAppAbout();
        DECLARE_MESSAGE_MAP()
};

extern CLab01App theApp;
```

Lab04.cpp:

```
// Lab04.cpp: определяет поведение классов для приложения.
//

#include "stdafx.h"
#include "afxwinappex.h"
#include "afxdialogex.h"
#include "Lab04.h"
#include "MainFrm.h"


#ifdef _DEBUG
#define new DEBUG_NEW
#endif
```

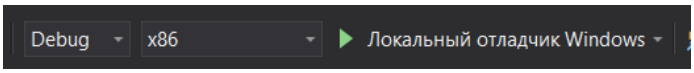
```
// CLab01App

BEGIN_MESSAGE_MAP(CLab01App, CWinApp)
END_MESSAGE_MAP()

// Создание CLab01App

CLab01App::CLab01App() noexcept
{

	SetAppID(_T("Lab01.AppID.NoVersion"));
}

// Единственный объект CLab01App

CLab01App theApp;


// Инициализация CLab01App

BOOL CLab01App::InitInstance()
{
	// InitCommonControlsEx() требуются для Windows XP, если манифест
	// приложения использует ComCtl32.dll версии 6 или более поздней версии для
включения
	// стилей отображения.  В противном случае будет возникать сбой при создании
любого окна.
	INITCOMMONCONTROLSEX InitCtrls;
	InitCtrls.dwSize = sizeof(InitCtrls);
	// Выберите этот параметр для включения всех общих классов управления, которые
необходимо использовать
	// в вашем приложении.
	InitCtrls.dwICC = ICC_WIN95_CLASSES;
	InitCommonControlsEx(&InitCtrls);

	CWinApp::InitInstance();


	// Инициализация библиотек OLE
	if (!AfxOleInit())
	{
		AfxMessageBox(IDP_OLE_INIT_FAILED);
		return FALSE;
	}

	AfxEnableControlContainer();

	EnableTaskbarInteraction(FALSE);

	// Для использования элемента управления RichEdit требуется метод
AfxInitRichEdit2()
	// AfxInitRichEdit2();

	// Стандартная инициализация
	// Если эти возможности не используются и необходимо уменьшить размер
	// конечного исполняемого файла, необходимо удалить из следующего
	// конкретные процедуры инициализации, которые не требуются
	// Измените раздел реестра, в котором хранятся параметры
	// TODO: следует изменить эту строку на что-нибудь подходящее,
	// например на название организации
	SetRegistryKey(_T("Локальные приложения, созданные с помощью мастера
приложений"));
```

```
	// Чтобы создать главное окно, этот код создает новый объект окна
	// рамки, а затем задает его как объект основного окна приложения
	CFrameWnd* pFrame = new CMainFrame;
	if (!pFrame)
		return FALSE;
	m_pMainWnd = pFrame;
	// создайте и загрузите рамку с его ресурсами
	pFrame->LoadFrame(IDR_MAINFRAME,
		WS_OVERLAPPEDWINDOW | FWS_ADDTOTITLE, nullptr,
		nullptr);


	// Разрешить использование расширенных символов в горячих клавишах меню
	CMFCToolBar::m_bExtCharTranslation = TRUE;




	// Одно и только одно окно было инициализировано, поэтому отобразите и обновите
его
	pFrame->ShowWindow(SW_SHOW);
	pFrame->UpdateWindow();
	return TRUE;
}

int CLab01App::ExitInstance()
{
	//TODO: обработайте дополнительные ресурсы, которые могли быть добавлены
	AfxOleTerm(FALSE);

	return CWinApp::ExitInstance();
}

// Обработчики сообщений CLab01App


// Диалоговое окно CAboutDlg используется для описания сведений о приложении

class CAboutDlg : public CDialogEx
{
public:
	CAboutDlg() noexcept;

// Данные диалогового окна
#ifdef AFX_DESIGN_TIME
	enum { IDD = IDD_ABOUTBOX };
#endif

protected:
	virtual void DoDataExchange(CDataExchange* pDX);    // поддержка DDX/DDV

// Реализация
protected:
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() noexcept : CDialogEx(IDD_ABOUTBOX)
{
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
```

```
{
	CDialogEx::DoDataExchange(pDX);
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialogEx)
END_MESSAGE_MAP()

// Команда приложения для запуска диалога
void CLab01App::OnAppAbout()
{
	CAboutDlg aboutDlg;
	aboutDlg.DoModal();
}

// Обработчики сообщений CLab01App
```

Теперь наше приложение готово к сборке.

Запускаем отладчик:



Получаем такой вот результат: